Downloaded from orbit.dtu.dk on: May 06, 2023

DTU Library

Taming Living Logic using Formal Methods

Baig, Hasan; Madsen, Jan

Published in:
Models, Algorithms, Logics and Tools

Link to article, DOI:
10.1007/978-3-319-63121-9_25

Publication date:
2017

Document Version
Peer reviewed version

Link back to DTU Orbit

Citation (APA):
Baig, H., & Madsen, J. (2017). Taming Living Logic using Formal Methods. In *Models, Algorithms, Logics and Tools* (pp. 503–515). Springer. Lecture Notes in Computer Science Vol. 10460 https://doi.org/10.1007/978-3-319-63121-9_25

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Taming Living Logic using Formal Methods

Hasan Baig and Jan Madsen

haba@dtu.dk, jama@dtu.dk
Department of Applied Mathematics and Computer Science,
Technical University of Denmark,
2800 Kongens Lyngby, Denmark

Abstract. One of the goals of synthetic biology is to build genetic circuits to control the behavior of a cell for different application domains, such as medical, environmental, and biotech. During the design process of genetic circuits, biologists are often interested in the probability of a system to work under different conditions. Since genetic circuits are noisy and stochastic in nature, the verification process becomes very complicated. The state space of stochastic genetic circuit models is usually too large to be handled by classical model checking techniques. Therefore, the verification of genetic circuit models is usually performed by the statistical approach of model checking. In this work, we present a workflow for checking genetic circuit models using a stochastic model checker (Uppaal) and a stochastic simulator (D-VASim). We demonstrate with experimentations that the proposed workflow is not only sufficient for the model checking of genetic circuits, but can also be used to design the genetic circuits with desired timings.

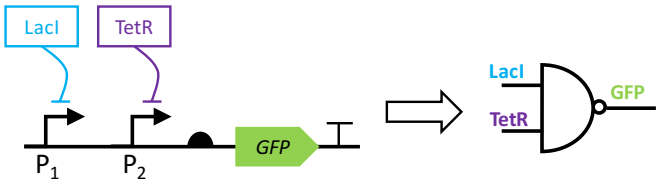
1 Introduction

Synthetic biology has emerged as an important discipline in which the synthetic digital [1, 2] and analog [2] computations in living cells have been implemented. Computation in living cells will revolutionize the fields of medicine and biotechnology. The aim of biological computation is to develop genetic devices to address the real-world problems including tumor destruction [4], bio-fuels [5], consuming toxic wastes [6], pharmaceuticals [7], etc. These biological devices are constructed from genetic circuits. A genetic circuit represents a gene regulatory network (GRN), which is composed of small genetic components, e.g., promoter, operator, ribosome binding site, protein coding site, and terminator. These components interact with the external signals (like temperature, light, etc.) to control the behavior of a living cell. Similar to electronic engineers who develop circuits using electronic logic gates (such as AND, NAND, and NOT gates), genetic network engineers use biological equivalents of these components to control the function of a cell [1][8].

Fig. 1(a) shows an example of a genetic implementation of a NAND gate represented in SBOL [9] notation. P_1 and P_2 are promoters, which are the regions of DNA that initiates the process of transcription (or production) of a gene. In this case, when two

proteins, LacI and TetR, are present in sufficient amount within the cell, they inhibit promoters P_1 and P_2 to produce the output protein i.e. green fluorescent protein (GFP). This type of gene regulatory networks is based on the "central dogma" of molecular biology, which states that genes in the DNA specify the sequence of messenger RNA (the transcription process by RNA polymerase), which in turn specify the sequence of proteins (the translation process by ribosomes). Regulatory proteins can control gene expression by either preventing transcription (repression), which is the case for LacI and TetR in the NAND gate, or by promoting RNA polymerase binding to the promoter (activation). A careful selection and balance of the genetic components, as expressed in the NAND gate in Fig. 1, can provide a functional gene regulatory network. To make genetic circuits work, it is not enough to be able to control the production of certain proteins, i.e. increasing the concentration, but also to be able to *reduce* concentrations of proteins. This happens by natural degradation of proteins, i.e. a protein has a certain lifetime, before it is dissolved into the amino acids from which it was constructed.

Signals in electronic logic gates propagate in separate electrical wires which do not interfere with each other, if designed correctly. However, in genetic circuits, signals are proteins drifting in the same volume of the cell, in order to establish a connection (a biological "wire"), compatibility between input- and output-proteins must be ensured and crosstalk with other signals from neighboring components, has to be avoided. This makes it challenging to work with genetic circuits, and thus requires a library of genetic components that can be used to develop complex circuits without causing crosstalk. The standard part libraries and toolboxes of well-characterized genetic components have been constructed through numerous laboratory experiments over the last decade [10-18]. These components have been extensively used to develop genetic circuits with different functionalities including oscillators [3], amplifiers [19, 20], linearizer gene circuit [21], memory devices [22, 23], switches [1, 12, 24], time-delay circuits [25,26], genetic logic gates [27-30] etc.



LacI	TetR	GFP
0	0	1
0	1	1
1	0	1
1	1	0

(c)

Fig. 1. Genetic NAND gate [55]. (a) Genetic implementation in SBOL notation. (b) Circuit schematic (c) Truth table.

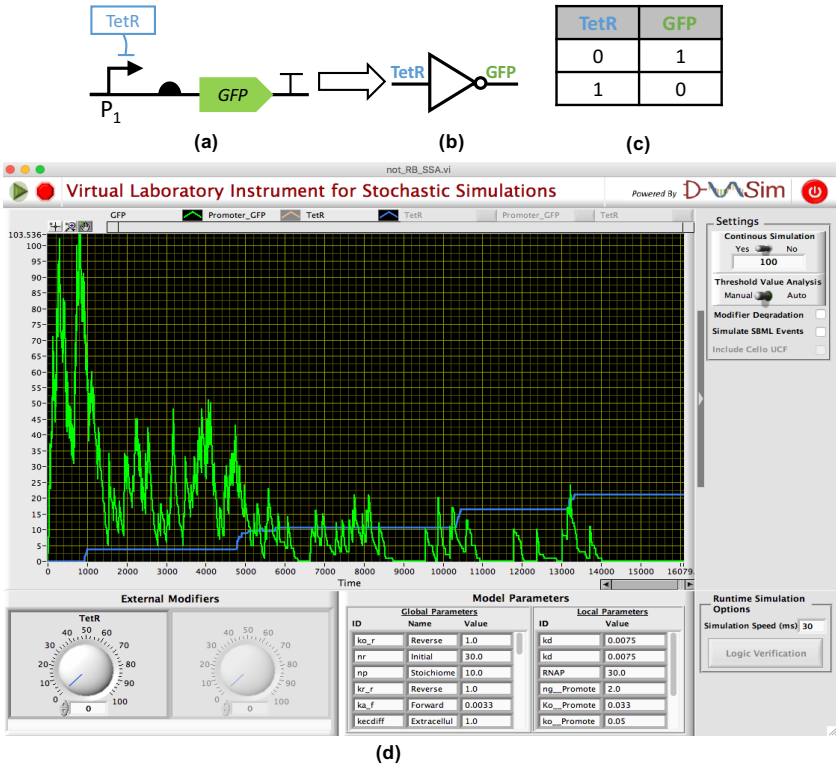
The field of synthetic biology is still in its infancy, and the process of design and implementation of genetic circuits remains very slow. Similar to the electronic design automation (EDA) process which dramatically enhanced the design, verification, validation and production of electronic circuits, researchers have started to work on the development of genetic design automation (GDA) tools to automate the design, test and verification processes of genetic circuits prior to their validation in laboratory. Several computational tools [31-34] have been developed to assist users in the model construction and design [35-37], simulation [35, 36, 38-40], logic and timing analysis of genetic circuits [40], and model checking [36, 41-44]. Model checking of biological systems is getting popular as it is an effective means of analyzing the dynamics of complex biological systems [45-53]. The dynamics of genetic circuits, and hence their correct functioning, are dependent on a large set of parameters (such as reaction and degradation rates) which in general are very difficult to predict and control. Hence, biologists are usually interested in determining the sensitivity of their circuits for fluctuations in these parameters. For instance, it might be a question of interest to find out, if the circuit behaves as expected when the values of certain parameters are varied within a specified range. Such sensitivity analysis is well suited for explorations using statistical model checking (SMC) and the aim of this work is to show how Uppaal SMC can be used to address the problem, effectively taming living logic.

In this work, we propose a flow of statistical model checking for genetic circuits using Uppaal [41] and D-VASim [39]. In particular, we performed experimentations on genetic circuit models and explored their design parameter sensitivity using Uppaal SMC [42]. There are a certain number of tasks which cannot be performed in Uppaal [41]. We therefore used D-VASim [39] to address those, which will be detailed in the experimentation section. The paper is organized as follows; Section 2 describes the digital abstraction and a brief introduction to D-VASim and Uppaal SMC. Section 3 contains the experimentation on genetic circuit models and Section 4 concludes the results.

2 Methodology

To determine the range of parameter values for which the genetic circuit would work, it is first important to know the threshold concentration levels of the inputs of those circuits. The threshold level of a genetic circuit can be defined as *the minimum concentration of input protein(s), which causes the average concentration of output protein to cross the level of input protein(s) concentration* [40]. D-VASim [39] is a simulation tool which supports the capability of analyzing the threshold value and timings of genetic circuits through an automated process. It further allows the user to perform runtime interactive simulations. For example, Fig. 2(d) shows the stochastic simulation traces of a genetic NOT gate obtained from D-VASim. The input is TetR protein and the output is GFP protein. When the input concentration of TetR goes high, the output concentration of GFP goes low.

In Fig. 2(d), the initial output concentration is about 100 molecules when the input concentration of TetR protein is low. When the concentration of TetR is triggered to 4 molecules, the concentration of output protein starts to degrade, but stays above the input concentration level. Increasing the input concentration further up (10 molecules) causes the output concentration to oscillate around the input concentration level. When we increase the input concentration level further (21 molecules), the output concentration stops oscillating around the input concentration level and settles down to zero. Here, the first input concentration level (up to 4 molecules) can be considered as low-threshold level as it does not cause the output concentration level to fall below it. Similarly, the third input concentration level (21 or more molecules) can be considered as high-threshold level as it causes the output concentration level to be in a clear logic-low state. The region between these two levels is considered as a transition region. This behavior is analogous to electronic circuits where the logic levels are well characterized. For example, the logic-1 of a 3.3V CMOS-based electronic device is at least 2.4V, which means that a minimum of 2.4V is required to turn the circuit on. Similarly, the circuit is considered off, when the output voltage is below 0.8V. The region between 0.8V and 2.4V is considered as a transition region, where the output is considered invalid.



TetR	GFP
0	1
1	0

Fig. 2. Genetic inverter (NOT) gate. (a) Genetic implementation in SBOL notation. (b) Circuit schematic. (c) Truth table. (d) Stochastic simulation traces in D-VASim.

Once the correct threshold levels are found, the inputs are triggered to that level and the circuit parameters can be varied to determine if the circuit still behaves correctly. As shown in Fig. 2(d), the threshold value and the logic of a circuit can be determined by varying the input concentration level and check if it significantly effects the concentration level of output. The case discussed above is a very simple case in which the genetic circuit has only one input and one output. However, this analysis could be very time consuming for large genetic circuit models with more inputs. For large-scale circuits, it is difficult to determine or verify the expected logic of a circuit without careful analysis. To determine or verify the logic of a genetic circuit, it is important to know the correct input combination with the correct threshold levels which trigger the output of the circuit. This may apparently become a tedious task to check different input concentration levels for each input combination.

The search process of threshold value can be automated by the use of statistical model checking in Uppaal. Uppaal is an integrated tool environment for modeling, verification and validation of real-time systems modeled as networks of timed automata. Uppaal SMC is an extended plug-in tool to Uppaal which allows the user to check the expected behavior of models in the form of probability distributions. In Uppaal SMC, it is possible to let the tool arbitrarily select any input concentration value, within a specified range, and see if the chosen value significantly effects the output concentration level. This can, however, only be achieved when the correct input combinations triggering the output of the circuit are known. As Uppaal does not have the capability to automatically detect the input combination which triggers the output of the circuit, the threshold value analysis of a genetic circuit cannot be performed *automatically* in Uppaal. D-VASim [39] is the only tool which allow users to perform threshold value and propagation delay analysis of genetic circuits through an automated process [40]. However, D-VASim is not capable of performing the automated statistical model checking. Thus, we used D-VASim for threshold value analysis and then perform the statistical model checking in Uppaal to determine the range of circuit parameters within which the circuit satisfy the desired behavior.

The proposed experimental flow of checking genetic circuit models is shown in Fig. 3. The genetic circuit models developed in the systems biology markup language (SBML) [53] are used in this work. The SBML model of a genetic circuit is used as input to D-VASim. D-VASim analyses the threshold and propagation delay (details are given in Section 3). The threshold value is then used in Uppaal to trigger the input levels to this value and observe the output behavior of the circuit while varying the circuit parameters. The effects of varying parameters on the threshold value and propagation delay of the circuit are then analyzed in D-VASim.

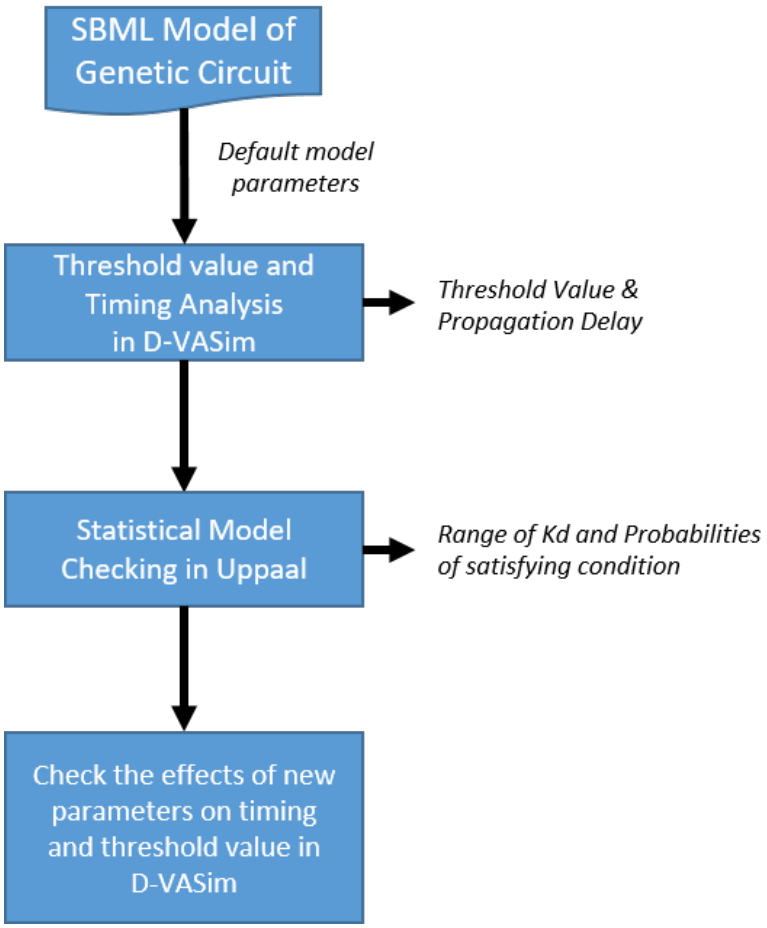


Fig. 3. Experimental flow of genetic circuit model checking and verification.

3 Experimentation

In this work, we test genetic circuit models from [55], by varying the degradation rate parameter (Kd) to determine the range within which the circuit exhibits the expected behavior. The aim is to propose an experimental flow for model checking of genetic circuits. To demonstrate that this flow can be applied to a complex genetic circuit as well, we have included the experimental results of a small (NAND gate) and a reasonably large (toggle switch with memory) genetic circuit models. The NAND gate contains 5 species and 5 kinetic reactions, whereas the toggle switch contains 20 species and its behavior is defined by 18 kinetic reactions. The schematic circuit models of the NAND gate and the toggle switch are shown in Fig. 1 and Fig. 4, respectively. In Fig. 4, the input protein A suppresses promoter P_1 to produce protein D, which in turn inhibits promoter P_4 to reduce the production of protein F, and so on.

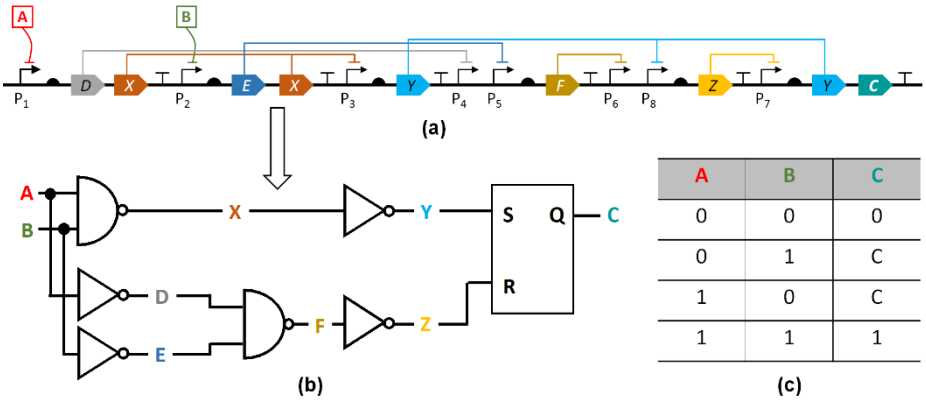


Fig. 4. Genetic toggle switch with memory [55]. (a) Genetic implementation in SBOL notation. (b) Circuit schematic. (c) Truth table.

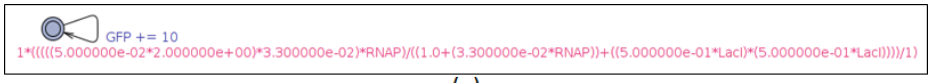
Table 1 shows the threshold and propagation delay values for both of the circuits obtained from D-VASim. The high threshold value specifies the input concentration level above which the logic is considered high, and the low threshold value specify the input concentration level below which the logic is considered low. The propagation delay is defined as *the time from when the input concentration reaches its threshold value until the corresponding output concentration crosses the same threshold value* [40]. The confidence intervals of threshold values are not specified in this table because D-VASim analyzes threshold values for pre-defined intervals of concentrations. For example, in the case of genetic NAND gate, the threshold level is analyzed for predefined concentration intervals each of which have a difference of 5 molecules. Therefore, D-VASim gradually increases the concentration from 0 → 5 → 10 → 15 and so on, to determine the lower and upper threshold levels of a NAND gate. For more accurate results, the concentration intervals for these analyses can be minimized in D-VASim.

Table 1. Threshold and propagation delay values obtained in D-VASim prior to SMC in Uppaal.

Circuit name	Threshold value (High)	Threshold value (low)	Propagation delay value
NAND	15	5	324 (±51.61)
Toggle Switch	10	5	1108 (±272.89)

These models are then checked in Uppaal by randomly choosing the value of Kd within a certain interval and checking if the output of the circuit satisfy the expected behavior for all possible input combinations. Uppaal uses a continuous time markov chain model (CTMC) for model checking, therefore the SBML models were first converted into CTMC models using the simple conversion utility in Uppaal. It creates a separate automaton for each of the reaction kinetics defined in the SBML file. For instance, Fig. 5(a) shows one of the processes, in the genetic NAND gate circuit, which

represents the kinetic reaction (Fig. 5(b)) to produce the 10 molecules of GFP when the input protein LacI is not sufficiently present in the cell.

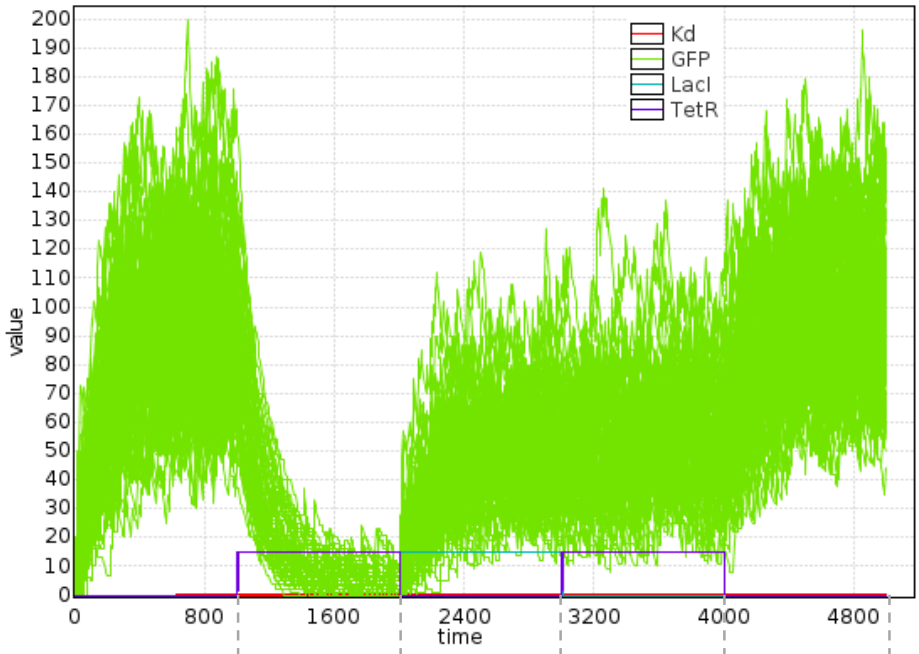


(a)

koP1*ngP1*KoP1*RNAP/(1+KoP1*RNAP+(KrLacIP1*LacI)^ncLacIP1)

(b)

Fig. 5. The process of a genetic NAND gate to produce the 10 molecules of GFP when the input LacI is not present in a cell (a) Uppaal interpretation. (b) Kinetic Reaction. Note that the value of ncLacIP1 is 2, due to which the factor KrLacIP1 is multiplied twice in (a).



Input combinations:	**11**	**10**	**01**	**00**
Probability:	0.92-0.99	0.95-1	0.95-1	0.95-1
Satisfying Simulations:	98 %	100 %	100 %	100 %

Fig. 6. Statistical model checking of the genetic NAND gate in Uppaal.

Fig. 6 and 7 shows the Uppaal SMC simulation results of the genetic NAND and the toggle switch circuits, respectively. These figures show all the simulation traces for 100 iterations. All possible input combinations are applied and the correct operation is verified within a defined range of Kd. Due to the stochastic nature of a model, the proba-

bility of an expected behavior cannot be 100% satisfied when the value of Kd is randomly chosen from a defined range. We, therefore, set the probability of the expected behavior to be greater than at least 90% as the acceptance criteria. Inputs correspond to the applied combination of input proteins over the course of simulation time. The logic-1 for the NAND gate corresponds to 15 or more molecules and logic-0 corresponds to 5 or less molecules. For the toggle switch, the logic-1 corresponds to 20 or more molecules and logic-0 corresponds to 10 or less molecules, as obtained from D-VASim.

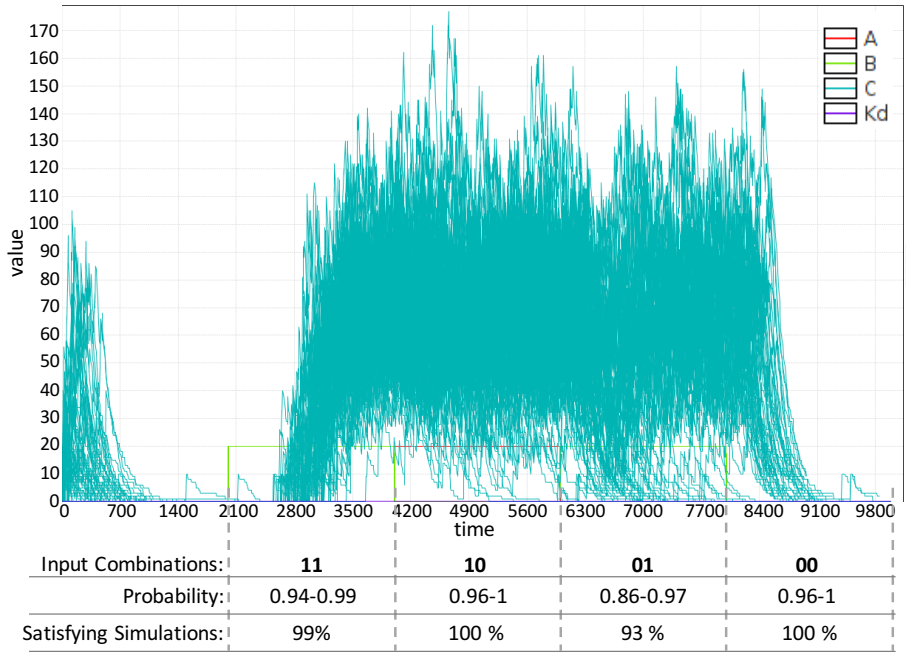


Fig. 7. Statistical model checking of the genetic toggle switch in Uppaal.

Probability values at the bottom of both figures signifies the probability of the expected behavior of a circuit for all possible input combinations, where each input combination is applied for 1000 time units for the NAND gate and 2000 time units for the toggle switch. These values are chosen sufficiently larger than their respective propagation delay values, estimated from D-VASim, to ensure that the appropriate amount of delay is provided to observe the effects of applied input combinations on the output of the circuit.

Satisfying Simulations indicates the percentage of simulations which satisfy the defined condition for specific input combination. These conditions are set according to the truth tables of respective circuits. For example, for the NAND gate, the condition to be checked for when the input combination is 11, is to see if the concentration of output protein, GFP, falls below its lower threshold level i.e. 5 molecules. The NAND gate circuit exhibits the probability of greater than 98% to work correctly when the

value of Kd varies between $45x10^{-4}$ and $85\ x10^{-4}$. Similarly, the toggle switch is at least 93% probable to work correctly when the value of Kd varies between $60x10^{-4}$ and $85x10^{-4}$. Outside, these ranges of Kd, the expected behavior do not satisfy the acceptance criteria mentioned above. In a similar manner, other circuit parameters can be varied to check the output response of genetic circuits.

Table 2. Threshold and propagation delay values obtained in D-VASim for upper and lower bounds of Kd values found in Uppaal.

Circuit name	Kd ($x10^{-4}$)	Threshold value (High)	Threshold value (low)	Propagation delay value
NAND	45	20	10	554 (±56.07)
	85	15	0	274 (±91.78)
Toggle Switch	60	20	10	1228 (±135.11)
	85	10	5	833(±97.41)

Finally, we used D-VASim to observe how the changes of Kd values impact the threshold value and the output of a circuit. In Table 2, we show the effects of the boundary values of Kd for both circuits. For example, in the case of the NAND gate, the effects of lower and higher-bound values of a Kd, $45x10^{-4}$ and $85\ x10^{-4}$, respectively, are checked. It is observed that the upper threshold concentration level required to trigger the output of the NAND gate is increased from 15 to 20 molecules when the value of Kd was decreased from $75x10^{-4}$ (default value) to $45x10^{-4}$. An increment in the propagation delay value is also observed. The latter is due to the fact that a decrease in the degradation rate causes the output response of the circuit to be slower, and thus more input concentration may be required to trigger the output. If the threshold value of a circuit is kept to its previous value, i.e., 15 at Kd = $45x10^{-4}$, the output may appear after a very long time; in other words, the propagation delay increases further. Likewise, when the value of Kd is increased to $85x10^{-4}$, the threshold values as well as the propagation delays are decreased. Similar observations have been made for the toggle switch as shown in Table 2. These observations indicate the minimum-high and maximum-low threshold values. For example, in order for the toggle switch to work within a range of Kd between $60\ x10^{-4}$ and $85\ x10^{-4}$, the minimum-high threshold value would be 20 molecules and a maximum-low threshold value would be 10 molecules.

4 Conclusion

In this paper, we propose a workflow for checking genetic circuit models using statistical model checking and stochastic simulation. We performed experimentations on two different-sized genetic circuit models to demonstrate that the proposed workflow can be applied for the timing and threshold values analysis of any genetic circuit model. We varied the design parameters of the genetic circuits and checked their probabilities

of working correctly. Furthermore, we analyzed the effects of changing design parameters on the behavior of a given circuit. The proposed work flow can be used to check any other property of a genetic circuit; such as the probability of a circuit to reach a certain state within a specific amount of time. Future work includes using the work flow to experiment with models of recently published genetic circuits [37] and to verify those results directly in the laboratory.

Acknowledgment

We would like to thank Marius Mikucionis (Aalborg University) for providing us an extensive support and help on using Uppaal. We would further like to thank Prof. Chris Myers (University of Utah) for providing us the SBML models of the genetic circuits, and Associate Prof. Michael Reichhardt Hansen (Technical University of Denmark) for fruitful discussions on model checking and for giving constructive feedback.

References

1. Gardner, T. S., Cantor, C. R. & Collins, J. J.: Construction of a genetic toggle switch in Escherichia coli. *Nature*, 403, 339–342, 2000.
2. Weiss, R. & Basu, S.: The device physics of cellular logic gates. *The First Workshop on Non-Silicon Computing*, 54–61, 2002.
3. Elowitz, M. B. & Leibler, S.: A synthetic oscillatory network of transcriptional regulators. *Nature*, 403, 335–338, 2000.
4. J C. Anderson et al.: Environmentally controlled invasion of cancer cells by engineering bacteria". *J. Mol. Biol.*, 355, pp. 619-627, 2006.
5. S. Atsumi and J. C. Liao: Metabolic engineering for advanced biofuels production from Escherichia coli. *Curr. Opin. Biotech.*, 19, 5, pp. 414-419, 2008.
6. I Cases and V De Lorenzo: Genetically modified organisms for the environment: stories of success and failure and what we have learned from them. *Int. Microbiol.*, 8, pp. 213-222, 2005.
7. Ro, D.-K. et al.: Production of the antimalarial drug precursor artemisinic acid in engineered yeast. *Nature*, 440, 940–943, 2006.
8. HH McAdams, L Shapiro: Circuit Simulation of Genetic Networks", *Science*, 269, 650-656, 1995.
9. B. Bartley *et al.*: Synthetic Biology Open Language (SBOL) version 2.0.0", *J. Integrative Bioinformat.*, vol. 12, no. 2, 2015.
10. Tom Ellis et al.: Diversity-based, model-guided construction of synthetic gene networks with predicted functions. *Nature Biotech.*, 27, 465-471, 2009.
11. Robert Sydney et al.: Programming gene expression with combinatorial promoters. *Molecular Systems Biology*, 3, 145*,* 2007.
12. Barry Canton et al.: Refinement and standardization of synthetic biological parts and devices", *Nature Biotech.*, 26, 788-793, 2008.
13. Mads Kaern et al., "The engineering of genetic regulatory networks", *Annu Rev biomed Eng.*, 5, 179-206, 2003.
14. Tom Knight: Idempotent vector design for standard assembly of biobricks", *MIT Artificial intelligence laboratory*, 2003. http://hdl.handle.net/1721.1/21168.

15. Salis, H.M., Mirsky, E.A. & Voigt, C.A.: Automated design of synthetic ribosome binding sites to control protein expression. *Nat. Biotechnol.* 27, 946–950, 2009.
16. Mutalik, V.K. et al.: Precise and reliable gene expression via standard transcription and translation initiation elements. *Nat. Methods,* 10, 354–360, 2013.
17. Cambray, G. et al.: Measurement and modeling of intrinsic transcription terminators. *Nucleic Acids Res.,* 41, 5139–5148, 2013.
18. Rodrigo, G. & Jaramillo, A. AutoBioCAD: Full biodesign automation of genetic circuits. *ACS Synth. Biol.,* 2, 230–236. 2013.
19. David Krig and Ron Weiss: Signal-amplifying genetic circuit enables in vivo observation of weak promoter activation in the Rhl quorum sensing system. *Biotechnol Bioeng.*, 89(6), 709-718*,* 2005.
20. Nistala GJ et al.: A modular positive feedback-based gene amplifier", *J Biol Eng.*, 4(1), 4, 2010.
21. Nevozhay D et al.: Negative autoregulation linearizes the dose-response and supresses the heterogeneity of gene expression. *Proc Natl Acad Sci USA*, 106 (13), 5123-5128, 2009.
22. Caroline M. Ajo-Franklin et al.: Rational design of memory in eukaryotic cells. *Genes Dev.*, 21(18), 2271-2276, 2007.
23. Fritz G, Buchler N, Hwa T, Gerland U: Designing sequential transcription logic: a simple genetic circuit for conditional memory. *Syst Synth Biol., 1*, 89–98, 2007.
24. Cherry JL and Adler FR: How to make a biological switch. *J Theor Biol*, 203 (2), 117-133*,* 2000.
25. Weber W et al.: A genetic time-delay circuitry in mammalian cell. *Biotechnol Bioeng.*, 98 (4), 894-902, 2007.
26. Caleb J Bashor et al.: Using engineered scaffold interactions to reshape MAP kinase pathway signalling dynamics. *Science*, 319, 5869, 1539-1543, 2008.
27. Guet CC, Elowitz MB, Hsing W, Leibler S: Combinatorial synthesis of genetic networks. *Science*, 296: 1466–1470, 2002.
28. Dueber JE, Yeh BJ, Chak K, LimWA: Reprogramming control of an allosteric signaling switch through modular recombination. *Science*, 301: 1904–1908, 2003.
29. Anderson JC, Voigt CA, Arkin AP: Environmental signal integration by a modular AND gate. *Mol Syst Biol*., 3: 133, 2007.
30. Win MN, Smolke CD: Higher-order cellular information processing with synthetic RNA devices. *Science*, 322: 456–460, 2008.
31. MacDonald, J. T., Barnes, C., Kitney, R. I., Freemont, P. S. & Stan, G.-B. V.: Computational design approaches and tools for synthetic biology. *Integr. Biol*., 3, 97–108, 2011.
32. Chandran, D., Bergmann, F. T., Sauro, H. M. & Densmore, D.: Design and Analysis of Biomolecular Circuits: Engineering Approaches to Systems and Synthetic Biology. *Springer*, 203–224, 2011.
33. Beal, J., Lu, T. & Weiss, R.: Automatic compilation from high-level biologically-oriented programming language to genetic regulatory networks. *PLoS ONE*, 6, e22490, 2011.
34. http://sbml.org/SBML_Software_Guide/SBML_Software_Matrix
35. Funahashi, A *et al.*: CellDesigner 3.5: A Versatile Modeling Tool for Biochemical Networks. *Proceedings of the IEEE*, 96, 1254 – 1265, 2008.
36. Myers, C. J. *et al.*: iBioSim: A tool for the analysis and design of genetic circuits. *Bioinformatics*, 25, pp. 2848–2849, 2009.
37. AA Nielsen *et al.*: Genetic circuit design automation. *Science*, vol. 352, issue 6281, 2016.
38. Hoops Stefan *et al.*: COPASI – a COmplex PAthway SImulator. *Bioinformatics*, 22, 3067-3074, 2006.

39. Baig, H. and Madsen, J.: D-VASim – An Interactive Virtual Laboratory Environment for the Simulation and Analysis of Genetic Circuits. *Bioinformatics*, vol. 32, no. 20, 1-3, 2016.
40. Baig, H. and Madsen, J.: Logic and Timing Analysis of Genetic Logic Circuits using D-VASim. *8th IWBDA 2016*, pp. 77-78, 2016.
41. Johan Bengtsson et al.: UPPAAL – a tool suite for automatic verification of real-time systems. *Proc 4th DIMACS workshop on verification and control of hybrid systems*, pp 232-243, 1995.
42. Peter Bulychev, Alexandre David, Kim G. Larsen, Axel Legay, Marius Mikucionis, and Danny Bøgsted Poulsen: Checking & Distributing Statistical Model Checking. *4th NASA Formal Methods Symposium*, *LNCS 7226, Springer,* pp 449-463, 2012.
43. Sumit Kumar Jha, Edmund M. Clarke, Christopher James Langmead, Axel Legay, André Platzer, and Paolo Zuliani: A Bayesian Approach to Model Checking Biological Systems. *In Proceedings of CMSB, LNCS 5688, Springer*, pp 218-234, 2009.
44. Edmund M. Clarke, James R. Faeder, Christopher James Langmead, Leonard A. Harris, Sumit Kumar Jha, and Axel Legay: Statistical Model Checking in BioLab: Applications to the Automated Analysis of T-Cell Receptor Signaling Pathway. *In Proceedings of CMSB*, *LNCS, Springer*, pages 231-250, 2008.
45. Calder, M., Gilmore, S., Hillston, J.: Modelling the influence of RKIP on the ERK signaling pathway using the stochastic process algebra PEPA. *Transactions on Computational Systems Biology*, VII, 4230, 1-23, 2006.
46. Calder, M., Vyshemirsky, V., Gilbert, D., Orton, R.: Analysis of signaling pathways using the PRISM model checker. *In: Proc. Computational Methods in Systems Biology (CMSB)*, pp. 179–190, 2005.
47. Cardelli, L.: Abstract machines of systems biology. *Transactions on Computational Systems Biology III. LNCS (LNBI), Springer, Heidelberg*, vol. 3737, pp. 145–168, 2005.
48. Fisher, J., Piterman, N., Hubbard, E.J., Stern, M.J., Harel, D.: Computational insights into caenorhabditis elegans vulval development. *Proc. Natl. Acad. Sci. USA*, 102(6), 1951–1956, 2005.
49. L. Calzone, N. Chabrier-Rivier, F. Fages, and S. Soliman: Machine learning bio-chemical networks from temporal logic properties. *Transactions on Computational Systems Biology VI (LNCS)*, *Springer, Berlin, Heidelberg,* 4220:68-94, 2006.
50. N. Chabrier and F. Fages.: Symbolic Model Checking of Biochemical Networks. *Proc 1st Internl Workshop on Computational Methods in Systems Biology*, pp 149-162, 2003.
51. M. Kwiatkowska, G. Norman, D. Parker, O. Tymchyshyn, J. Heath, and E. Gaffney.: Simulation and verification for computational modelling of signaling pathways. *WSC '06: Proceedings of the 38th conference on Winter simulation*, pp 1666-1674, 2006.
52. C. Langmead and S. K. Jha. Predicting protein folding kinetics via model checking. *Lecture Notes in Bioinformatics: The 7th Workshop on Algorithms in Bioinformatics (WABI)*, pp 252-264, 2007.
53. C. Langmead and S. K. Jha. Symbolic approaches to finding control strategies in boolean networks. *Proceedings of The Sixth Asia-Pacific Bioinformatics Conference, (APBC),* pp 307-319, 2008.
54. The Systems Biology Markup Language (SBML): Language Specification for Level 3 Version 1 Core, October 06, 2010.
55. Chris J. Myers.: Engineering Genetic Circuits. *Chapman & Hall/CRC Press*, 2009.